УДК 519.876.2(65.011.56)

СОПОСТАВЛЕНИЕ ТРАДИЦИОННЫХ МЕТОДОЛОГИЙ ОПИСАНИЯ БИЗНЕС-ПРОЦЕССОВ И ЯЗЫКА ИХ ИСПОЛНЕНИЯ

И.Г. Озерова

Томский политехнический университет
E-mail: ira@aics.ru

Предложено преобразование описаний бизнес-процессов, выполненных с помощью традиционных методологий, например, DFD (Data Flow Diagram) в язык BPEL (Business Process Execution Language), для чего выполнено сопоставление этих методологий. Получены правила разработки схемы бизнес-процесса на основе диаграммы потоков данных, конструкциям которой поставлены в соответствие элементы языка BPEL.

Для получения дополнительного конкурентного преимущества современные предприятия внедряют решения на основе систем управления бизнес-процессами (*Business Process Management, BPM*). Одним из компонентов *BPM*-системы является графический редактор (дизайнер). Он обеспечивает визуальную разработку (описание, проектирование) схем бизнес-процессов, что можно сравнить с работой в привычных средах моделирования, таких как *AllFusion Process Modeler* (ранее *BPwin*), *ARIS Toolset.* Графической схеме соответствует описание на специальном языке, которое для исполнения загружается в движок *BPM*-системы. Исполненные бизнес-процессы могут быть проанализированы с помощью модуля мониторинга. В настоящее время процесс стандартизации языков, используемых в *BPM*-системах, не завершен, но уже приняты базовые спецификации, определяющие правила построения и выполнения бизнес-процессов и способы их взаимодействия с программной средой [1]. На сегодняшний день наиболее перспективным *BPM*-стандартом, на который ориентируются все ведущие производители программных продуктов и технологий, является язык *BPEL* [2]. Стандарт *BPEL*, разрабатываемый с 2002 г., включает в себя средства для организации согласованной работы (оркестровки – *orchestration*) нескольких приложений и обмена сообщениями между ними. В данной статье используется именно язык *BPEL*.

Начальный этап разработки бизнес-процесса – этап анализа и проектирования. Несмотря на то, что *BPM*-система включает в себя графический редактор, был сделан вывод, что для качественного анализа и проектирования бизнес-процессов более приемлемо использование традиционных средств моделирования и анализа, как то: *BPwin*, который поддерживает *IDEF0*, *DFD*, *IDEF3*; *ARIS Toolset* (в данной статье рассматривается методология *DFD*). Если же предприятие ранее уже описало свои бизнес-процессы в виде диаграмм с помощью указанных средств в ходе выполнения соответствующего проекта, то при внедрении *BPM*-системы этап анализа и проектирования пропускается, а из описанных бизнес-процессов выбираются те, которые требуется автоматизировать средствами *BPM*.

Реализация бизнес-процесса заключается в создании его описания на языке *BPEL* (кодировании), которое выполняется с помощью визуального (графического) редактора, что практически полностью избавляет от необходимости непосредственно писать код. Поэтому предлагается выполнять реализацию в два подэтапа:

- выполнить построение в графическом редакторе *BPEL* на основе модели, полученной на этапе анализа и проектирования, т. е. преобразовать модель *BPwin* в *BPEL* (для такого преобразования необходимо выработать правила);
- создать необходимые сообщения, переменные, определить их тип, назначить соответствующим блокам и т. п.

Чтобы получить правила преобразования из *DFD* в *BPEL*, необходимо выделить характерные конструкции диаграммы *DFD* и поставить им в соответствие конструкции *BPEL.*

Конструкциями диаграммы *DFD* являются:

- блок (функция, *activity*), декомпозиция;
- поток данных, слияние и ветвление стрелок;
- внешняя сущность;
- хранилище.

Далее необходимо найти способ описать эти конструкции средствами *BPEL.* В данном исследовании используется *BPM*-система *Oracle BPEL Process Manager.*

Блок *DFD*, являющий собой в модели какое либо действие, в *BPEL*, видимо, также будет действием, описанным с помощью какого-либо блока, в зависимости от контекста.

Блок *DFD*, подлежащий декомпозиции, детализируется на нижнем уровне иерархии несколькими блоками. Для группировки нескольких блоков в *BPEL* существует два компонента: `scope` и `sequence`. `Scope` (один из блоков на рис. 1) – это контейнер, состоящий из других вложенных блоков произвольной глубины, которые могут иметь свои собственные локальные переменные, обработчики ошибок и т. д. `Scope` является аналогом операторных скобок в языках программирования. Использование блока `scope` упрощает *BPEL*-поток путем группировки функциональных структур и возмож-

ности свернуть их в один элемент [3]. То же справедливо и для `sequence` с той разницей, что этот элемент не имеет локальные переменные, обработчики ошибок и т. д. Поэтому блок, для которого в *DFD* существует диаграмма-потомок, следует заменить на блок *BPEL* `scope` или `sequence` с сохранением имени (*name*) блока. Это, с одной стороны, позволит избежать необозримости диаграммы *BPEL*; с другой стороны (для `scope`), даст возможность ввести локальные переменные и обработчики ошибок. Если блок `scope` включает в себя более одного блока, то внутри него создается блок `sequence`, и вложенные блоки будут заключены уже в этот блок, который позволяет определить совокупность блоков, выполняемых последовательно. Все переменные, созданные для контекстной диаграммы, являются глобальными, а обработчики ошибок не используются на этом уровне, поэтому блок контекстной диаграммы *DFD* всегда описывается блоком `sequence` в *BPEL*.

Как правило, бизнес-процесс инициируется поступлением запроса от внешней сущности (например, клиента). Этот поток изображается на контекстной диаграмме *DFD* в виде стрелки, направленной от внешней сущности к блоку. В *BPEL*, чтобы сохранить полученные данные в переменную для дальнейшего использования, необходимо включить блок `receive`, ожидающий сообщение от внешнего сервиса (*Partner Link*), которое сохраняется в соответствующей переменной процесса. Таким образом, первым элементом диаграммы *BPEL* будет `receive` (рис. 1).

Если клиент, инициировавший бизнес-процесс, должен получить ответ, то в *DFD* это изображается в виде стрелки, направленной от процесса к внешней сущности. Чтобы отразить это на диаграмме *BPEL*, необходимо использовать блок `reply` или `invoke` для синхронного или асинхронного бизнес-процесса соответственно. Оба блока передают партнеру (внешней сущности, сервису) сообщение, которое хранится в переменной, закрепленной за данным блоком.

Синхронный *web*-сервис обеспечивает немедленный ответ на запрос. Сервер *BPEL* позволяет задать максимальное время ожидания синхронного ответа, значение которого по умолчанию 60 с. Если за это время ответ не получен, происходит отказ. Асинхронный способ обмена сообщениями приме-

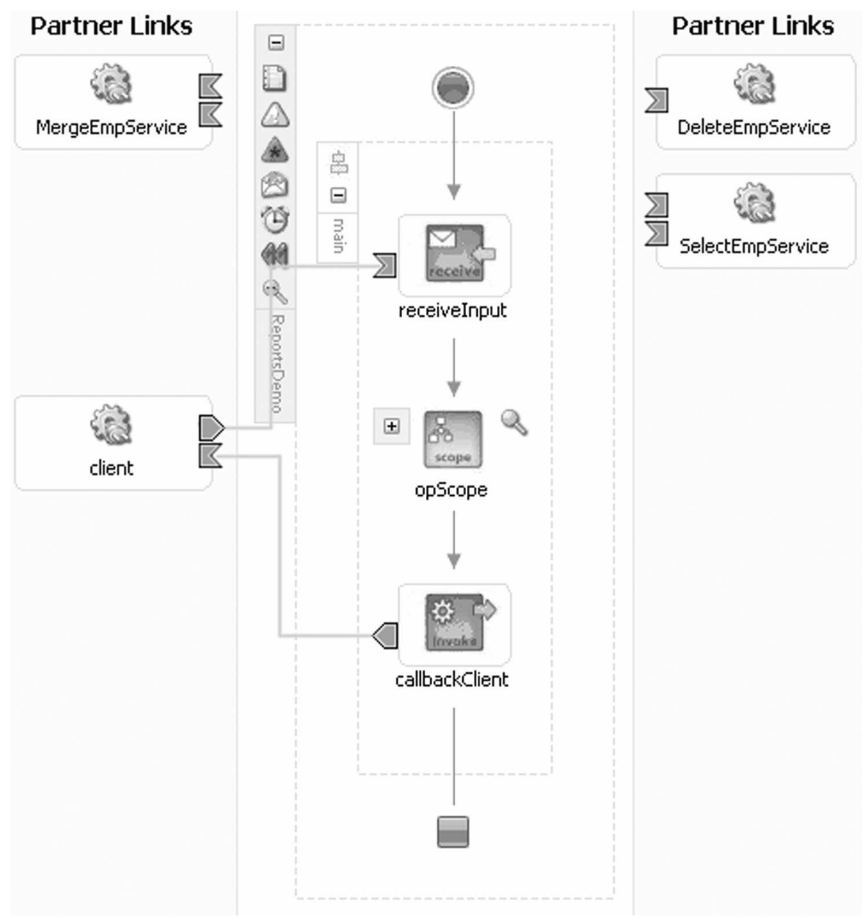


Рис. 1. *Диаграмма BPEL*

няется, когда, например, сервис может потребовать длительного времени обработки заявки клиента (от нескольких минут до нескольких дней). Асинхронный сервис обеспечивает более надежную, отказоустойчивую и масштабируемую архитектуру. Таким образом, чтобы клиент бизнес-процесса получил ответ, необходимо следующее:

- выбрать способ обмена сообщениями (оценить время обработки заявки);
- если это синхронный процесс, добавить на диаграмму *BPEL* в качестве последнего процесса блок `reply`, или асинхронный – `invoke`;
- соединить добавленный блок с внешней сущностью (сервисом, *Partner Link*), если она уже добавлена на диаграмму.

Если на вход блока *DFD* поступают данные (например, данные клиента), являющиеся результатом слияния других данных от разных источников (например, фамилия клиента и его электронный адрес, рис. 2), то с точки зрения *BPEL* это значит, что существуют три переменные, содержащие фамилию, адрес и данные в совокупности, а для выполнения функции блока необходима переменная, в которой будут храниться данные клиента.



Рис. 2. *BPwin. Слияние данных*

Для того, чтобы записать данные клиента в эту переменную, требуется использовать блок `assign`, который позволяет выполнять манипуляции с данными, например, конкатенацию или копирование содержимого одной переменной в другую, в данном примере – содержимое исходных двух переменных в соответствующие части конечной. Например, если конечная переменная имеет имя `clientRequest`, то ее содержимое после копирования может иметь следующий вид:

```
<clientRequest>
   <part name=»payload» >
      <ClientDataProcessResponse>
         <lastName>Ivanov</lastName>
         <email>ivanov@sibmail.ru</email>
      </ClientDataProcessResponse>
   </part>
</clientRequest>
```

Таким образом, для выполнения слияния потоков данных в *BPEL* необходимо ввести блок `assign` с соответствующим(и) правилом(ами) копирования (*copy rule*(*s*)), которые создаются в окне указанного блока (рис. 3).

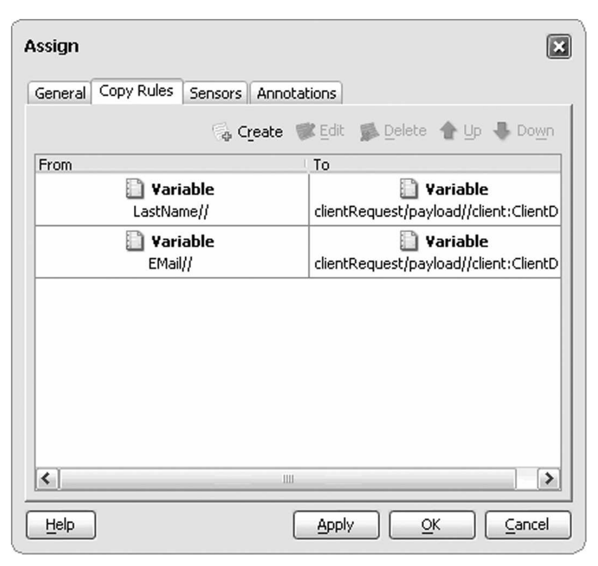


Рис. 3. *BPEL. Окно блока* `assign`

Ветвление стрелок в *BPwin* отображается на диаграмме *BPEL* аналогичным образом с помощью блока `assign`: в две (или более) переменные с помощью *copy rule* копируются соответствующие части исходной переменной (исходных данных).

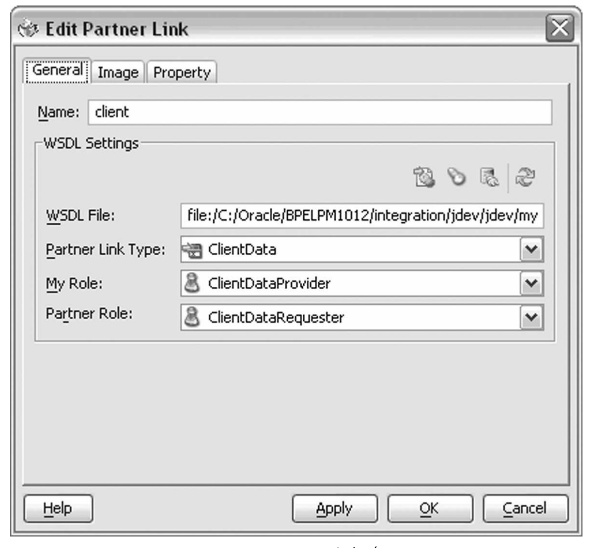


Рис. 4. *Редактирование Partner Link (в том числе установка ролей)*

Внешняя сущность на диаграмме *DFD* является источником/приемником данных. Но что бы она не представляла собой, это находится за границами моделируемого бизнес-процесса, который не рассматривает внутреннюю работу внешней сущности. *BPEL* как раз и предназначен для оркестровки таких внешних сущностей. На диаграмме *BPEL* они представлены в виде элементов *Partner Link*, которые могут также быть *BPEL*-процессом или любым другим *web*-сервисом, с которым взаимодействует данный. Для каждого *Partner Link* определяются роли бизнес-процесса, в котором используется сервис, и его самого (*Requester* – запрашивающая сторона и *Provider* – поставщик инфор-

мации, услуг, рис. 4). Если требуется реализовать бизнес-процесс таким образом, что он инициируется по запросу клиента, например, с *web*-сайта, то на диаграмме *BPEL* клиент будет представлен в виде соответствующего *Partner Link*, так как бизнес-процесс взаимодействует с клиентом как с внешним сервисом. Причем роль *Partner Link* (клиента) будет *Requester*, а роль бизнес-процесса – *Provider*. Если в ходе выполнения происходит обращение к другому внешнему сервису за информацией, то роль *Provider* будет играть внешний сервис.

Общение функции *DFD* с хранилищем представляется в виде запрос-ответ, причем ответ возвращается немедленно, без предварительных дополнительных вычислений. Поэтому на диаграмме *BPEL* хранилищу из диаграммы *DFD* необходимо поставить в соответствие *Partner Link*, который, в свою очередь, является синхронным сервисом взаимодействия с базой данных. В более редких случаях хранилище может быть не базой данных, а например, бумажным архивом, работа с которым выполняется человеком и требует времени. Тогда на диаграмме *BPEL* следует использовать *Partner Link* (асинхронный сервис), которому может соответствовать *web*-страница с запросом и полями для ответа. Понимание того, синхронным или асинхронным будет взаимодействие с хранилищем, важно, т. к. требуется соответственно один или два типа порта (*port type*). Тип порта – это совокупность связанных операций, выполняемых участником во время диалога. Тип порта определяет, какая информация посылается и принимается, форму этой информации и т. д. Синхронный обратный вызов требует только один тип порта, который и посылает запрос, и получает ответ, в то время как асинхронный обратный вызов (где ответ не немедленный) требует два типа порта: один для того, чтобы послать запрос, а другой – чтобы получить ответ, когда он придет.

Стоит отметить, что для правильного преобразования модели из *DFD* и *BPEL* необходимо хорошо знать контекст бизнес-процесса, что видно на примере преобразования хранилища.

Таким образом, для более эффективной разработки бизнес-процессов было выполнено сопоставление традиционных методологий и языка исполнения. Показано, что схема *BPEL* может быть разработана путем анализа и проектирования бизнес-процесса с помощью *BPwin* (*DFD*) и последующей реализации с использованием полученных в результате сопоставления правил преобразования конструкций *DFD* в *BPEL*.

СПИСОК ЛИТЕРАТУРЫ

1. Свинарев С. Business Process Management: от идеи к практической реализации // PC WEEK. – 2005. – № 34. – С. 39–41.
2. Гайдай А. На пути к BPEL. – 2004. – http://www.oracle.com/global/ru/oramag/dec2004/gen_bpel_it.html [07.09.2006].
3. Oracle BPEL Process Manager Developer's Guide, 10g Release 2 (10.1.2). – 2005. – 566 p. – http://download.oracle.com/otndocs/products/bpel/bpeldev.pdf [11.09.2006].